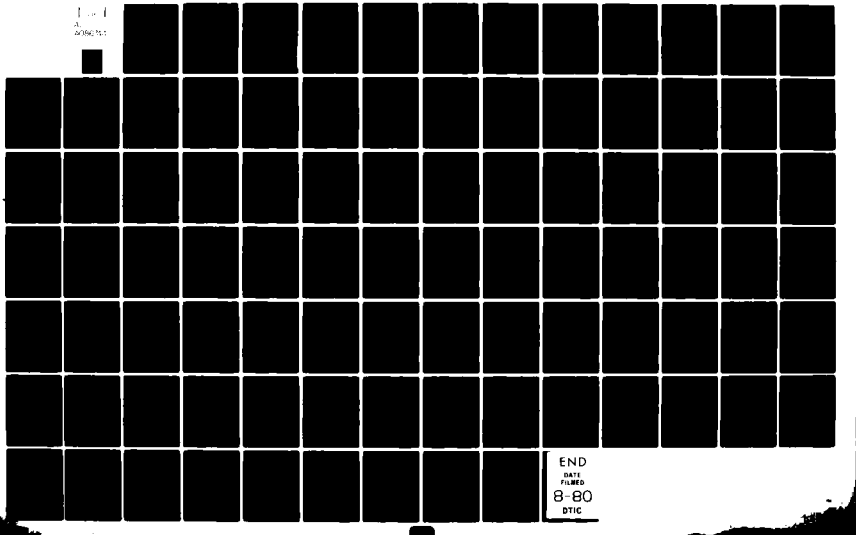
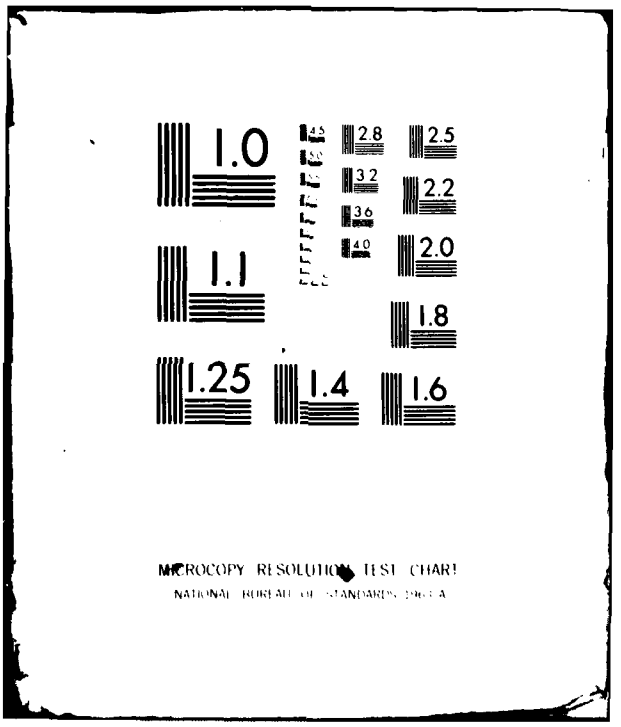
AD-A086 744 ARMY MILITARY PERSONNEL CENTER ALEXANDRIA VA F/G 12/2
MULTIPLE OBJECTIVES AND THE PATH DETERMINATION PROBLEM.(U)
JUL 80 J E THOMAS

UNCLASSIFIED NL



END
DATE FILMED
8-80
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

MULTIPLE OBJECTIVES AND THE PATH DETERMINATION PROBLEM

CPT Joseph E. Thomas, Jr.
HQDA, MILPERCEN (DAPC-OPR-E)
200 Stovall Street
Alexandria, VA 22332

Final Report, 3 July 1980

A thesis submitted to The Johns Hopkins University in partial fulfillment of the requirements for the degree of Master of Science in Engineering.

LEVEL II

MULTIPLE OBJECTIVES AND THE PATH DETERMINATION PROBLEM

by

Joseph E. Thomas, Jr.

An essay submitted to The Johns Hopkins University in partial fulfillment of the requirements for the degree of Master of Science in Engineering.

Baltimore, Maryland

1980

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A086 744 — 3. RECIPIENT'S CATALOG NUMBER (9)
4. TITLE (and Subtitle) (6) MULTIPLE OBJECTIVES AND THE PATH DETERMINATION PROBLEM	5. TYPE OF REPORT & PERIOD COVERED Final report, 3 July 1980
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (10) Joseph E. Thomas, Jr.	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Student, HQDA, MILPERCEN (DAPC-OPP-E) 200 Stovall Street Alexandria, VA 22332	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS HQDA, MILPERCEN, ATTN: DAPC-OPP-E 200 Stovall Street Alexandria, VA 22332 (11)	12. REPORT DATE 3 July 1980 — 13. NUMBER OF PAGES (12) 90
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Thesis submitted to The Johns Hopkins University in partial fulfillment of the requirements for the degree of Master of Science in Engineering.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Networks; Graphs; Shortest path labeling algorithms; Multiobjective analysis; Multiple criteria analysis; Network flow; Noninferior path labeling Algorithm

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The shortest path problem exists as a major component of many network problems. For problems with a single objective, algorithms developed over the past twenty-five years have made the solution of this problem a relatively simple exercise. For multiobjective problems, adaptation of shortest path methodologies to generate noninferior solutions is a complex and difficult task. This thesis introduces an algorithm, based on label-correcting algorithmic techniques, which identifies all noninferior paths from one node in a network to all other nodes in the network in a single computer run. (Continued on reverse)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

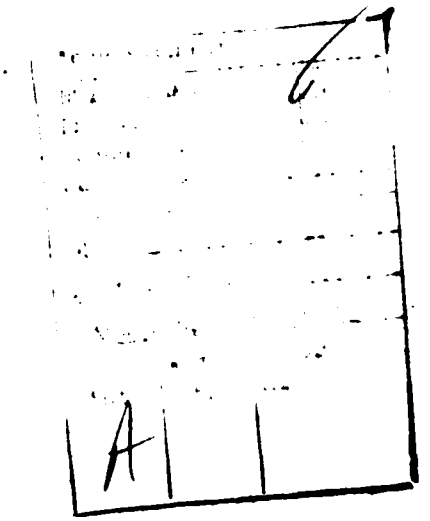
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

291191

20. (Cont,)
The algorithm, titled the Noninferior Path Labeling Algorithm (NPLA), is shown to be at least four to ten times faster than shortest path labeling algorithms applied multiobjectively for a problem with two objectives. NPLA is judged to be unique in its capability to identify all solutions which lie in duality gaps of the noninferior solution set. NPLA has been found to be easily programmed, efficient, and capable of being used with all network designs. It is readily adaptable to problems of more than two objectives.

ABSTRACT

The shortest path problem exists as a major component of many network and network-related problems. For problems having a single objective, algorithm development over the past twenty-five years has made the solution of this problem a relatively simple exercise. For multiobjective problems, adaptation of shortest path methodologies to generate noninferior solutions becomes a complex and difficult task. This thesis introduces an algorithm, based upon standard label-correcting algorithm techniques, which identifies all non-inferior paths from one node in a network to all other nodes in the network in a single computer run. The algorithm, titled the Non-inferior Path Labeling Algorithm (NPLA), is shown to be at least four to ten times faster than shortest path algorithms applied multiobjectively for a problem with two objectives. NPLA is judged to be unique in its capability to identify all solutions which lie in duality gaps of the noninferior solution set. NPLA has been found to be easily programmed, efficient, and capable of being used with all network designs. It is readily adaptable to problems of more than two objectives.



ACKNOWLEDGEMENTS

The author would like to express his appreciation to all those who helped make this thesis possible, in particular to:

Professors Jared L. Cohon and Charles S. ReVelle, who gave me more than generous advice and encouragement.

The U.S. Military Academy and the U.S. Department of the Army which entrusted me with time and opportunity.

The U.S. Department of Energy for its essential support which made this research effort possible.

Colleagues Donald Shobrys and Roger Cox for the former's able direction and the latter's exceptional programming and computational contributions.

To my wife, Ann, for her support, encouragement, and forbearance, and my daughter, Christine, whose smile dissipated my discouragements.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd.)

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

PATH DETERMINATION AND MULTIOBJECTIVE ANALYSIS

A. INTRODUCTION

This thesis presents the results of research directed toward identification and implementation of a multiobjective methodology for the determination of paths through a network. The path-determination, or routing, problem lies at the heart of many public and private system planning issues. Mail delivery, refuse collection, and police patrol design are everyday examples in which the routing problem is a central focus. The use of an existing transportation system and the design of additions to a transportation system require the application of path-determination methodologies by both public authorities and private enterprise planners. Pipeline systems, water supply systems, communication systems, electronic systems design, aircraft routing, and the routing of shipments of toxic substances are other examples where the routing problem is an important issue. In addition to the areas where routing is a central focus, path-determination is an essential component of many methodologies designed to locate facilities in a network. These include the p-Median Problem, Warehouse Location Problem, the Transportation Problem, and Network Flow Problems.

Within the area of path-determination, the problem of finding

the shortest path between nodes within a network is often of prime importance. Although it is difficult to underestimate the importance of the route-determination problem itself, the degree of algorithm development over the past twenty-five years has made the solution of the Shortest Path Problem a relatively straightforward and simple process. Such cannot be said, however, when the path-determination problem involves multiple objectives. Instead of finding a single, shortest path out of many alternative paths between two nodes, the multiobjective path-determination problem requires the determination of one or more noninferior paths between any two nodes in a network. In such cases, the "shortest" path will be one of the noninferior paths, but, very often, it will not be an unambiguously superior path.

Multiple objectives are a common element of many problems facing planners in all elements of public and private operations. Planners are increasingly turning to multiobjective analytic methodologies for their solutions. The application of multiobjective analysis to the path-determination problem has been rare, but the requirement for such analysis is appearing and its importance increasing.

B. MULTIOBJECTIVE ROUTING PROBLEM FORMULATION

This thesis presents the results of research directed toward the identification and implementation of a multiobjective methodology for the path-determination problem. This problem arose as a result of a research project in the Department of Geography and Environmental Engineering of The Johns Hopkins University, funded by the U.S. Department of Energy (DOE). The purpose of the DOE project was to develop a prescriptive model for the location of Away-From-Reactor (AFR) spent nuclear fuel storage facilities. A network approach was chosen for the modeling effort. Four tasks were required to be undertaken as follows:

1) Define the network.
2) Determine which AFR sites should be open to accept spent fuel.
3) Determine which reactors will ship their spent fuel to which of the open AFR's.
4) Determine the appropriate routes over which spent fuel will be shipped between any given reactor and AFR.

Network definition consisted of selecting nodes to represent potential AFR sites and existing reactor sites, selecting the system of arcs and nodes representing the transportation system to be used, and determining and assigning traversal costs to each arc. Solution

of the AFR opening and reactor assignment problems was approached through the application of a p-Median methodology. [D. Shobrys, Unpublished Memorandum, Johns Hopkins University, 1980 and DOE/SR Interim Report, Johns Hopkins University, 1980.] The approach to the routing problem was not immediately decided upon and was the subject of careful research.

Care in the selection of a route-determination methodology was important for two principal reasons. First, the p-Median methodology requires the predetermination of routes between all AFR-reactor pairs. Traversal costs for these routes form the cost vector of the p-Median problem formulation. Its solution, AFR openings and reactor assignments, is a reflection of, and therefore dependent upon, the routes which have been determined. The second reason is the attention paid by DOE and the public to the issue of nuclear waste shipments. Since proximity to the nuclear wastes appears to be a principal concern, the choice of routes and how they are chosen must be subjects of close examination. This last consideration motivated the use of a multiobjective approach to the problem.

There are efficient and relatively simple methodologies available for solving the path-determination problem with respect to a single objective, which is typically cost, distance, or time minimization. However, the DOE project was to look at two objectives, distance minimization and minimization of population within a

specified zone along the routes. The first objective is a surrogate for cost while the second provides a measure of the perceived undesirability of shipping spent fuel over the route [D. Shobrys, Unpublished Memorandum, The Johns Hopkins University, 1980, and DOE/SR Interim Report, The Johns Hopkins University, 1980].

Although there are many problems for which multiple objectives can be identified, this problem in particular is an example in which a single objective would incompletely describe the problem and most likely be unacceptable in a political decision-making arena. The two objectives chosen here represent diverse opinions of what is important about nuclear waste shipping routes and AFR location.

The routing problem was now required to be formulated so as to permit the generation of the noninferior set of paths between each potential AFR site and reactor in the network. Relatively few techniques have been developed for generating noninferior sets [Cohon, 97]. With the exception of the multiobjective simplex method, generating methods obtain solutions iteratively from a transformed, single-objective version of the problem. In this manner, the multiobjective path-determination problem can be reduced to a single-objective problem by creating weighted combinations of the objectives. Run many times using an appropriate single-objective methodology and a different weight for each run, many different solutions may be found. Those solutions which are noninferior can

be plotted in objective space to form a representation, to a greater or lesser degree, of the complete set of noninferior solutions.

During the initial stages of the DOE project, the multiobjective routing problem was approached in this way. However, as a result of investigation, an algorithm was developed by the author and project analyst Roger Cox which is capable of generating the complete set of noninferior paths between any two points of a network in a single run. The algorithm, titled the Noninferior Path Labeling Algorithm (NPLA), has served to reduce much of the difficulty associated with the multiobjective path-determination problem.

C. THESIS ORGANIZATION

This chapter has established the routing problem, discussed its difficulties when approached multiobjectively, and briefly introduced an approach to facilitate its solution. The remaining chapters serve to more thoroughly examine single- and multiobjective path determination methodologies and explain the NPLA.

Chapter II discusses the history and present status of shortest path algorithms. Also discussed are existing algorithmic approaches to multiobjective path determination, the K-Shortest Path approach in particular.

Chapter III formally presents NPLA. It examines its relationship to single-objective shortest path algorithms, states the algorithmic

form, and illustrates its workings and capabilities by example.

Chapter IV examines the relative efficiencies and capabilities of NPLA with respect to other multiobjective approaches. The results are based on data obtained from the DOE project.

Chapter V offers conclusions about the present status of NPLA and directions for future research.

CHAPTER II

THE SHORTEST PATH PROBLEM AND MULTIOBJECTIVE APPLICATIONS

A. INTRODUCTION

The most basic, and one of the most common, of the path-determination problems is the shortest path problem. In its most general form the problem may be stated as: find the shortest path between specified pairs of nodes in a network. In modified form, the problem statement may be amended to: find the noninferior path(s) between specified pairs of nodes. This is the essence of multiobjective path-determination. It will be shown in Chapter III that NPLA has many structural and operational similarities to some shortest path algorithms. This chapter is designed to place NPLA in the context of the development and structure of shortest path algorithms. It is further examined with respect to other multiobjective approaches based on shortest path algorithms.

The organization of this chapter reflects an ability to place shortest path algorithms in various categories. One can identify two principle methodological approaches applicable to the problem of determining shortest paths: those based on linear programming and those based on the fields of graph theory and combinatorics. Section B is devoted to the linear programming based algorithms. These algorithms play a relatively small part in practical problems of

shortest path determination. With the exception of their role in multiobjective analysis, discussion of these algorithms is limited to this section.

Section C begins the discussion of shortest path algorithms based on graph theory and combinatorics. It is devoted to the history of their development and a review of the pertinent literature, with a goal of placing this area in chronological perspective.

Section D continues the discussion about algorithms based on graph theory and combinatorics. In this section, the shortest path problem statement has been restated to show two distinct subproblems: 1) find shortest paths from all nodes to all other nodes (all-to-all) and 2) find shortest paths from one node to another node (one-to-one). Consistent with these two categories there exist two categories of algorithms to achieve these goals. Section D discusses matrix algorithms to solve the "all-to-all" problem and introduces the class of labeling algorithms which solve the one-to-one problem.

Section E is devoted to a detailed discussion of labeling algorithms, the most generally efficient and commonly used set of shortest path algorithms. Labeling algorithms can also be categorized in two ways: as label-correcting and label-setting algorithms. This section established the notation and concepts of graph theory necessary for the understanding and operation of labeling algorithms, and presents both classes of labeling algorithms.

Section F concludes the chapter with a discussion of approaches to multiobjective path-determination. Existing examples have been drawn from the literature. Included is an illustration of how a multiobjective noninferior set generation technique can be applied with shortest path algorithms.

B. NETWORK FLOWS AND SHORTEST PATHS

The first expression of a network flow problem was the Transportation Problem. Here an inventory must be moved from a specified number of sources to a specified number of sinks. The path from each source to each sink is presented as though it were a single link. Consistent with the origin of the Transportation Problem, paths and costs are often associated with existing transportation routes and rates as expressed by commercial water, road, rail and air freight charts. Algorithmic approaches to the transportation problem were supplemented by the linear programming technique by Dantzig in 1951, [see Bazaraa and Jarvis, 68] beginning an association of network flow and linear programming theories.

Recognizing the essentially heuristic nature of determining links and costs in the Transportation Problem, the Transshipment Problem was formulated by Alex Orden in 1956 [4]. Orden's formulation permitted the existence of a network consisting of intermediate nodes and arcs as well as source and sink nodes. He was able to restructure

this network into the Transportation Problem format. In this way, the problem of routing was formally addressed, along with the problem of assigning shipping from sources to demand points. The problem could still be solved by the existing Transportation Problem solution techniques. Orden went one step further, however, by recognizing that the routing capability of the Transshipment Problem formulation could address the shortest path problem. He proposed that his formulation be used to obtain the link cost data for direct use in large transportation problems.

In the years after Orden's work, much was accomplished in network flow theory. Included in this development was the definition of the class of problems called Minimum Cost Network Flows and development of versatile and efficient solution algorithms. Again the concentration is on routing an inventory through a network, but additional constraints are placed on the network in the form of upper and lower bounds on link capacities, and the structural restrictions of the Transportation/Transshipment format were eliminated. Though the relationship to the original Transportation Problem is clear, the emphasis of the Minimum Cost Network Flow problem has clearly shifted to include route selection.

As in the Transshipment Problem, the Minimum Cost Network Flow Problem can be structured to determine shortest paths [Bazaraa and Jarvis, 68]. Unlike the Transshipment Problem, there exist many

solution techniques for the Minimum Cost Network Flow Problem. These techniques -- primal, dual, primal-dual and its related out-of-kilter algorithms among them -- all reflect the continuing association of network flow and linear programming theory.

Both the Transshipment Problem and Minimum Cost Network Flow Problem offer several advantages to an analyst in search of a shortest path solution technique. There exist a number of proven algorithms and program codes [68, 80, 81, 82, 83, 89, 90, 91]. The literature offers the results of detailed testing of program codes on a wide range of networks [84, 85, 86, 87, 88, 92]. Their linear programming basis permits the use of parametric and sensitivity analysis [68, 93, 96]. Recently, there has appeared a number of additions to the literature which illustrate experience in the application of multiobjective methodology to these problem areas [93, 94, 98, 99, 100, 101].

The basic disadvantage to the entire network flow area lies in the acknowledged, marked inefficiency of any network flow solution technique in comparison to a set of algorithmic techniques devised specifically to solve the shortest path problem. So great is the disparity, that the routing capability of network flow algorithms is rarely used to solve the shortest path problem.

C. GRAPHS, COMBINATORICS, AND SHORTEST PATHS

The second methodological approach applicable to shortest paths is based in the fields of Graph Theory and combinatorics. The use of the term shortest path algorithm usually refers to a technique associated with this approach. A graph is a representation of an existing or theoretical physical system. Graph theory is concerned with graph structure, design, and physical capabilities and limitations. Shortest paths, shortest spanning trees, and shortest circuits, including the Traveling Salesman Problem, are associated with Graph Theory. Graphs are distinguished from networks only by the absence of inventory and flow considerations, with Network Flow Theory being primarily concerned with flow optimization through a given network. This paper will continue to use "network" to describe the node-arc system. Its commonly understood meaning is a sufficiently accurate representation of graphs and networks as used in practical applications of Graph and Network Flow Theory.

Most shortest path algorithms are combinatorial in nature. Since networks consist of a finite set of discrete nodes and arcs, a path or route through a network can be viewed as one particular combination of node-arc linkages of a finite, discrete, countable set of possible combinations. Such a set could be found by exhaustive search. The shortest path could be discerned from this set but at prohibitive cost for any non-trivial network. The algorithms discussed

below serve to conduct an efficient search of node-arc combinations by confining the search to stages at which only a subset of the set of nodes and arcs are examined. Implicitly or explicitly, these algorithms use Bellman's dynamic programming principle of optimality [69], in paraphrase: A subpolicy of an optimal policy must itself be optimal.

History: Tracing the earliest development of shortest path algorithms is a difficult process. Early work in shortest paths was confined to heuristic methods which have not survived in practice. Some influence on shortest paths appears to have been exerted by the work on shortest spanning trees [see Shimbel, 3] for reasons which should become clear in discussion below. Later history is confused due to conflicting references in the literature, probably due to the geographical and disciplinary dispersion of authors of shortest path algorithms and the obscurity of the original publications containing algorithms.

The first algorithmic formulation of the shortest path problem was apparently offered by Ford in 1956 [2], the same year Orden published his Transshipment Problem. Unfortunately, it first appeared in a proprietory document, the subject of which was Network Flow Theory. Although later published in France in a collection by Berge [10], it received little recognition and was further obscured by later, independently derived algorithms.

In the years afterward, many shortest path algorithms were presented. In 1957, Dantzig [5] presented an algorithm similar in nature to Ford's, and Mintz [6] offered an analog method. Moore [7] offered several algorithms in 1957 but his contribution did not receive wide recognition until publication in 1959. In 1958, Mintz [11] offered a formal algorithm and Bellman [9] presented several algorithms tied to his dynamic programming theory. Dijkstra [12] published a definitive algorithm in 1959 with Whiting and Hillier [18] publishing a similar algorithm in 1960. In 1962, Floyd [22] published his definitive algorithm as a one paragraph presentation in *Communication of the Association for Computing Machinery*. His recognition was also subject to delay until it was discovered and republished in 1965 by Murchland [32].

By 1965, it can be said that the basic formulations for the shortest path solution algorithms had been defined. Work thereafter has primarily been directed toward marginal algorithmic improvements and explorations into improvements in implementation. Contributors to the area of shortest paths included computer scientists, electrical engineers, dynamic programmers, network flow theorists, integer programmers, graph theorists, and transportation scientists. It is a striking note that the bulk of development was derived independently, and over a relatively short period, by people from such diverse disciplines.

Literature: The body of literature on the shortest path problem is extensive. Two excellent bibliographies (Pierce [63] in 1975 and Golden and Magnanti [70] in 1977) form the basis for a review of the literature. The selected bibliography presented here attempts to place shortest path development in chronological perspective and list as many of the articles pertaining to shortest paths, published after the above mentioned bibliographies, as were found during the course of this research.

Within the literature, there are many publications which are especially instructive to researchers and analysts. Many surveys have been published which detail shortest path algorithms, discuss theoretical efficiencies, and present the results of comparative computational studies of algorithms and associated implementation techniques [17, 20, 43, 44, 56, 60, 64, 72, 73, 78]. The confusion stemming from such a proliferation of algorithms is mitigated by the efforts of several articles to categorize the many algorithmic variations in terms of a small set of basic approaches, simple in concept [56, 64, 72, 78]. Other articles detail computer implementation and data processing techniques designed to increase the efficiency of the basic algorithms [15, 31, 36, 47, 48, 52, 55, 57, 58, 67, 71]. The state of the literature is such that an analyst may select an algorithm and computer coding directly or develop his own formulation based on his choice of a basic algorithm and his

programming experience.

D. SHORTEST PATH PROBLEM CATEGORIES AND ALGORITHMS

In examining the shortest path algorithms based on graph theory and combinatorics, a division into two general categories can be seen. The shortest path algorithms may be distinguished according to which of two problem statements they address:

1) Find the shortest path from some start node s to some end node t in a network (one-to-one).
2) Find the shortest path from every node in a network to every other node in a network (all-to-all).

This latter problem can be solved readily by the application of algorithms developed by Bellman (1959) [9], Tabourier (1973) [58], Floyd (1962) [22] and others [30, 33, 55]. These algorithms known as matrix algorithms are attractive when shortest paths are desired between a significant number of s-t node pairs. A principal disadvantage of matrix algorithms is the requirement for storage during program run, a requirement which makes problems of even moderate size impossible to be processed on a computer. This problem has been attacked through the development of decomposition procedures [37, 38, 42] to increase the size of the problem which can be successfully solved. A survey of these algorithms and computational experience is available [33]. The survey also compares the efficiency

of these algorithms with "one-to-one" shortest path algorithms repeated sufficiently to derive shortest paths from all nodes to all other nodes. The seeming inefficiency in this latter approach is reduced by two elements. First, the all-to-all algorithms are costlier to run than one-to-one algorithms. When it is actually desired to find those shortest paths between two proper subsets of the total set of nodes (corresponding to a physical network of origins and destinations with the remaining nodes being connecting or transshipment nodes) and one or the other of the two subsets is small in comparison with the total number of nodes, the one-to-one algorithms applied several times will tend to be more efficient than the all-to-all algorithms. Second, as will be further explained, one-to-one algorithms solve their problems by actually finding all shortest paths from a single node s to all other nodes. This means that, as an upper bound, a so-called "one-to-one" algorithm will solve the all-to-all problem when run N times for an N-node network. In the case where the network is too large to allow use of an "all-to-all" algorithm, use of the "one-to-one" algorithms remain the only option.

For the DOE project cited earlier, the network of N=447 nodes would require CPU storage on the order of N^2 (200,000) addresses to obtain a solution with an all-to-all algorithm. To extract the paths, however, additional storage on the order of N^3 (90,000,000)

addresses is required (Dreyfus [44]). Even with the possibility of structural decomposition, "all-to-all" algorithms represent an unattractive option for solving this problem.

The following section discusses the "one-to-one" problem of shortest paths and the algorithms which most efficiently attack this problem.

E. SHORTEST PATH LABELING ALGORITHMS

When examining the "one-to-one" shortest path problem, the firmest conclusion which can be drawn from the literature involves the choice of algorithmic technique. Although there are several approaches, including network flow algorithms, analogs [6, 17], and algebraic techniques [see 72], the class of algorithms known as labeling algorithms is acknowledged to be the most generally applicable, versatile, easily coded, and efficient. As such, labeling algorithms form the basis for the discussion in this section.

As a prerequisite to the discussion of labeling algorithms, a presentation of some notation and concepts is warranted.

Let $e_{ij} \equiv$ an arc beginning at node i and ending at node j, $i \neq j$ (let no arc exist which starts and ends at the same node). Note that a direction is implied. If a given network is physically undirected (two way traffic with costs equal in either direction), another arc e_{ji} must be designated or coding introduced to read the

arc as doubly directed.

A path *P* from *s* to *t* is a series of arcs, $P = (e_{si}, e_{ij}, \cdots e_{kl}, e_{lt})$, such that each arc in the sequence begins at the node that the previous arc reached. If s = t, the path starts and ends at the same point, and the path is called a *circuit*. The path is *simple* if each arc and node in the path appears only once. Note that there is no prohibition against having more than one arc, going in the same direction, between any pair of nodes. Although all shortest path algorithms will allow multiple arcs between two nodes, most physical networks will include one arc for any direction, that is selected for its shortest distance.

Paths may also be represented by a sequence of nodes *v*, where $P = v_s, v_1, \ldots, v_t)$ and node v_i is the end of the arc which begins at node v_{i-1} and node v_{i-1} is termed the predecessor of node v. Numbering of nodes may be accomplished arbitrarily, an exception being a node numbering system for acyclic networks (highly directional networks containing no possible circuits).

The *length* of an arc, l(e), is the arc's contribution to the attribute in question and may, therefore, be cost, distance, time, or any other measure of attribute level. The length of a path is the linear sum of the individual arc lengths of which the path is comprised. A shortest path is that path, out of the feasible paths between two given points, which is smallest. By reversing the

criteria, a longest path may also be determined. An undirected arc is symmetric if its length is the same in both directions.

A rooted tree is a directed network consisting of n-1 nodes and a root node s such that each of the n-1 nodes can be reached by one *simple* path from s. That is, every node v, (except s) is the end of only one arc (no restriction on the number of arcs beginning at v); no arc ends at s (no circuit); and, there are no circuits embedded in any of the paths. A rooted tree is a minimum tree, or shortest path tree, if for each node v in the network, the unique path from s to v is also a shortest path from s to v [see 56].

Given a representation of a physical network which is undirected and symmetrically weighted, such as a highway network, rooted trees may be constructed by removing (ignoring) arcs which violate the conditions for a tree (or minimum tree). It may not be possible to construct such a tree to include all nodes in a highly directed network, where travel between many pairs of nodes is restricted to one direction only, but such a concern does not normally apply to a highway network. If a minimum tree rooted at s can be constructed, then a shortest path between s and *any* node t has been found. Moreover, if what is actually desired is the shortest routes from a single node s to many nodes t_k, a procedure which constructs a rooted minimum tree presents a decided advantage.

The shortest path labeling algorithms developed to solve the

problem of finding a shortest path from a node s to a node t in a network have done so by constructing minimum trees rooted at s, thereby producing the more general result of finding shortest paths from s to all other nodes in the network. As discussed earlier, given a set A consisting of n origins and a set of B of m destinations, both A and B being subsets of the set of all nodes in the network, a shortest path labeling algorithm can find the $n \cdot m$ shortest paths desired in only n computer runs of the algorithm.

As an example, consider the AFR location problem. The network consisted of 447 nodes, 75 of which were sources and 24 of which were destination nodes. Since the network was both undirected and symmetrically weighted, the smaller subset of nodes (24) could be treated as the set of origin nodes for the purpose of structuring the data for the algorithm. Thus, there exist 75 shortest paths from each of 24 origins. These $24 \cdot 75 = 1800$ shortest paths which need to be determined could be found with 24 computer runs of a shortest path labeling algorithm.

Labeling algorithms get their name from the bookkeeping technique (label) associated with each network node and which is essential to the algorithms' operations. The node label is simply a number representing the shortest distance from the source to the labeled node found at a particular stage in an algorithm's calculations. As an algorithm proceeds, labels are changed and set until it has

been determined that no label for any node can be changed. At that point, the algorithms stop and all node labels represent the shortest path length from source to any other node. It remains to determine the arcs and nodes forming the path. In order to easily extract the actual path another number, a node predecessor, may be associated with each node label. The node predecessor is the number of the next node in the shortest path closer to the origin thus designating an arc in the shortest path. Extracting the path is then a matter of backtracking to the origin and recording the nodes or arcs in order, by hand or through additional coding.

Labeling algorithms have been described as falling into two general classes, label-correcting and label-setting [56, 72]. The distinction is based on when any particular node label can be declared permanent, that is, recognized as being the actual shortest path length. In label-correcting algorithms, no node label is declared permanent until, at the termination of the algorithm, all are declared permanent. In label-setting algorithms, node labels are periodically declared permanent as the algorithms proceed. The algorithms terminate when all node labels have been declared permanent. Other performance distinctions associated with these two basic approaches are examined below.

Label-correcting algorithms were initially described by Ford (1956) [2], Moore (1957) [7], and Bellman (1958) [9]. Excellent

discussions of the individual algorithms and the label-correcting algorithm as a class are given by Dreyfus [44], Gilsinn and Witzgall [56], Hulme and Wisniewski [72], and others [17, 64, 78]. The approach taken by these algorithms is to establish a tree rooted at the origin. Arcs of the basic network not currently in the tree are examined in an attempt to find some which can be inserted in the tree, in place of arcs already in the tree. The criterion used is that an exchange of arcs must result in a lower node label value. This method may be generally described as follows [56, 72].

Let d(i) - the label associated with node i

(1) Initially $d(s) = 0$ (s = origin)

$d(i)$ = distance from s to i along a path in the tree

$d(i) = \infty$ if node i is not connected to s by any path.

Any tree can be used to establish initial values of the d(i), $i \neq s$. However, it is sufficient to start with the tree consisting of node s alone and no arcs.

(2) Search through the list of arcs, not currently in the tree, for an arc e_{ij}, any $i \neq j$, such that the sum of the current node label at the predecessor node and the arc length is less than current node label at j ($d(i) + l(e_{ij})$ d(j)). If such an arc is found, place it in the tree, remove the arc that had previously been

attached to node j, and change the node label at j to $d(j) = d(i) + l(e_{ij})$ and the predecessor to i.

(3) Continue the search procedure until no arcs can be found to reduce any node label. Stop the algorithm. The tree is now a minimum tree rooted at s. For each node j, d(j) is the length of the shortest path from s to j.

The first label-setting algorithm is ascribed to G. Minty by Pollack and Wiebenson [17]. Others were described by Dantzig [15], Hu [45], and Whiting and Hillier [18]. The label-setting algorithm most often cited as being the most efficient was described by Dijkstra [12]. Label-setting algorithms start from the origin node and proceed to construct a minimum or shortest route tree step-by-step, stopping when the tree includes all nodes in the given network. It may be described as below [56, 72]:

(1) Let $d(s) = 0$

T ≡ shortest path tree, initially consisting of the origin, s, alone.

(2) Search through the arcs that begin at a node in T but end at one of the nodes which is not yet in T. Among those arcs e_{ij}, i in T, j not in T, find the one that minimizes $d(i) + l(e_{ij})$. (As opposed to any arc which would reduce a node label - as in label-correcting algorithms.) Place that arc and node in T, set or reset

$d(j) = d(i) + l(e_{ij})$, and declare it permanent ($d(j)$ is now the length of the shortest path from s to j).

(3) Repeat (2) until all nodes are in T and permanent.

The logic which permits label-setting algorithms to establish permanent node labels prior to termination of the algorithm requires that no arc weights be non-negative. The minimization element of the process in step (2) allows the node label to be declared permanent. The reasoning is that a more indirect route to node j would have to originate from one of the nodes which, from the minimization calculation, is already larger than the node label set at j. Call this other candidate node k. $d(k)$ is > (or at best =) $d(j)$ from minimization. Therefore, $d(k) + l(e_{kj}) \geq$ (equal if $l(e_{kj}) = 0$), *if* $l(e_{kj}) \geq 0$. Were $l(e_{kj})$ allowed to be less than 0, it is apparent that it is possible that $d(k) + l(e_{kj}) < d(j)$ and that (e_{kj}) should be included in T, e_{ij} removed from T and $d(j)$ changed to equal $d(k) + l(e_{kj})$. However, a label-setting algorithm, by declaring $d(j) = d(i) + (e_{ij})$ as permanent, would not find the better solution above, and the tree found by the algorithm would not be a shortest path tree.

In examining the two basic categories of labeling algorithms, theoretical bounds were calculated as to computational efficiencies based upon the number of additions and comparisons (e.g., $d(i) + l(e_{ij})$ as an add, $d(i) + l(e_{ij}) \lessgtr d(j)$ as a compare) required by each algorithm to construct a shortest path tree. In a comprehensive review of algorithms, including most of the initial formulations

and modifications known at the time, Dreyfus (1969) [44] calculated theoretical upper bounds on the number of calculations as follows: label-correcting algorithms were to have computer run times proportional to N^3 where N = number of nodes in the network; label-setting algorithms were to have run times proportional to N^2. On that basis, label-setting algorithms should be the preferred approach to solving shortest path trees, given a network such as a highway network where all arc weights are positive.

Two factors about the algorithms and the efficiency calculations resulted in further investigation into relative efficiencies. First, theoretical calculations were based on upper bounds which meant the assumption of a complete network: all nodes connected by an arc to all other nodes in the network. Physical networks rarely approach this representation and transportation networks are very sparse. A second factor involves modifications to the basic algorithms, many of which are stated in the literature as new algorithmic types. The modifications have been aimed at two elements within Step (2) of both label-setting and label-correcting algorithms. In both cases, the algorithms require in Step (2) a search of arcs. The order in which arcs are stored and are examined at each iteration in the search procedures can have an effect on the efficiency of an algorithm. Computational examinations were conducted on algorithms of both categories using different "list-processing" modifications to the basic algorithms and applied to networks of varying degrees of

arc-density. Other experiments investigated the effect of network "shape" (e.g., long and thin vs. square or circular) on algorithm efficiencies. The results of such experiments have been reported by several authors including Gilsinn and Witzgall (1973) [56] and Hulme and Wisniewski (1978) [72]. A general conclusion reached from the results of such experiments was that relative differences in efficiencies between the two classes of algorithms were reduced or even reversed, given certain list-processing modifications and networks with specific arc densities. Arc density is the measure of the completeness of a network. In a complete network every node is connected to every other node and has, therefore, N(N-1) (about N^2) arcs where N is the number of nodes in the network. Arc density is the ratio of the actual number of arcs in the network (n) to the maximum number of arcs and is normally calculated: n/N^2.

Hulme and Wisniewski [72] investigated three algorithms, using associated list-processing modifications and applying them to sparse networks, with arc densities between 0.5 and 0.001 and a number of nodes which varied between 25 and 2000. Hulme and Wisniewski chose two label-correcting algorithms: an algorithm by Ford [2] and independently described by Moore [7], and Yen [47]. The remaining algorithm was the label-setting algorithm described by Dijkstra [12] using a list-processing modification termed the heap sort. Their choice of algorithms was based on previously reported experiments.

The algorithms themselves are explained in detail and Fortran codings are offered as appendices. Their results indicate that Ford's label-correcting algorithm was superior to Yen's and to Dijkstra's label-setting algorithm in most cases.

Despite the extensive computational experience offered, an analyst searching for an algorithm for a particular network is still confronted with uncertainties in these results. Other considerations which affect performance are age and type of computer hardware, coding languages, and supplemental coding designed to organize and display the algorithm's results. In general, computational experience was gathered on a particular set of hardware, with a particular language, and with little attention given to procedures for presenting and displaying results. It is a warning to the analyst that computational results be taken as a guide to relative performance during calculations. The run times offered may be altered by hardware and language and dominated by subsequent data display coding.

F. SHORTEST PATHS AND MULTIPLE OBJECTIVES

The shortest path problem has not been examined with respect to its use in multiobjective analysis. This categorical statement may be argued against, but the literature reveals only cursory mention of multiobjective analysis and a methodological approach

which can be considered only marginally effective in multiobjective analysis.

Christofides in his 1975 book *Graph Theory: An Algorithmic Approach* [62] is the only reference found in the literature which talks of multiple objectives. The text outlines several examples of problems with two attributes. In all cases, however, an implicit judgement has been made which results in a commensuration of the attributes. An example is the problem of probabilistic networks. The outcome of the analysis is the shortest expected distance. While clearly a traditionally acceptable result, there exists the possibility of "shorter" but "riskier" paths and, conversely, "longer" but "more secure" paths. In the absence of any discussion, one must assume that a commonly accepted commensuration of attributes has been substituted for multiobjective analysis in the problem of probabilistic networks and similar problems.

Christofides does mention, but does not illustrate with an example, the use of a weighted combination of objectives when the attributes in question are not perceived to be obviously commensurable. Again, no discussion is forthcoming on choosing the appropriate weights, nor is there any reference to texts or articles about multiobjective analysis.

This is not to say that multiobjective analysis does not apply to shortest path problems. Christofides' discussion illustrates the

applicability of a preference-oriented multiobjective approach, a particular methodological approach that obviates the need for multiobjective shortest path algorithms. However, the mention of a weighted combination of objectives (the *only* mention in the shortest path literature) does suggest that the weighting method, one of the principal generating techniques may have a role to play.

There does exist an approach in the shortest path literature which offers explicit recognition of the existence of multiple objectives and their potential effect on the decision about which path would be considered best. This approach has become known as the K^{th} Shortest Path Problem. Stated briefly, the problem is to: find the shortest, 2^{nd} shortest, ..., k^{th} shortest paths between two given nodes in a network. Note that the K^{th} Shortest Path Problem does not address the question of multiple objectives. The premise is that, given the existence of an additional quantitative or qualitative attribute, the decision-maker can pick one among the k paths identified which possesses an acceptable combination of distance and whatever other attribute is of concern. That is, "shortest" is still defined in terms of the conventional distance or cost objective.

The K^{th} Shortest Path Problem has a long history. The first algorithm acknowledged to address the problem was stated by Hoffman and Pavley in 1959 [13] making it one of the earlier attempts

to address the question of multiple objectives. Also interesting is that it occurs early in the development of algorithms for the Shortest Path Problem as well. Other proposed algorithms quickly followed including one by Bellman and Kalaba in 1960 [14] and by Pollack in 1961 [19]. Surveys of K^{th} shortest path algorithms were included in articles by Pollack in 1961 [19] and Dreyfus in 1969 [44].

The attractiveness of K^{th} shortest path algorithms is severely diminished by their lack of direct consideration of any but a single objective to identify paths. As a result, none of these algorithms can assure that any but the shortest of the k paths identified will be noninferior with respect to the multiple objectives. Further, there is no assurance that all noninferior paths can be identified short of complete enumeration.

As a result, the K^{th} Shortest Path Problem remains an unsatisfactory approach to multiobjective analysis. Unfortunately, it appears to be the only approach to have been widely considered for multiobjective shortest path problems.

Although not discussed in the literature relating to shortest paths, it is clear that the weighting method of multiobjective analysis (see Cohon, 1978 [97]) can be used with shortest path algorithms to generate noninferior paths.

In the weighting method, a two-objective problem would be reduced to a single-objective problem in the following manner:

$$Z^W = wZ^1 + (1-w)Z^2$$

where Z^1 = objective one, Z^2 = objective two, w = weighting factor permitted to equal a value anywhere in the closed interval $[0,1]$, and Z^W = the weighted combination of the two original objectives.

At any choice of w, the problem is reduced to a single objective problem which can be run using any available solution technique. The resulting solution is the best solution given the relative value of the two objectives with respect to each other, as determined by the choice of w. The analyst proceeds to systematically vary the choice of w within the prescribed interval, thereby generating a succession of solutions, each of which can be expressed in terms of its contribution to the two attributes and the weight at which it can be described as "best". A complete discussion of the weighting method is available in Chapter 6, reference [97].

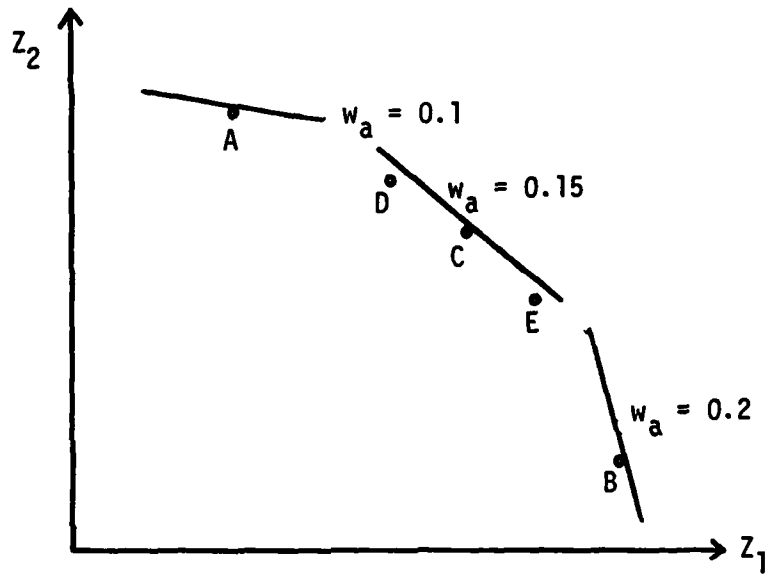
The weighting method is confronted with two major difficulties: one general to any application, the second created by the nature of networks. In general, any given noninferior solution to a multi-objective problem will be found by the weighting method for a single weight value but for a range of weight values (e.g., Solution A will be found whenever a weight, w, wuch that $w_1 \leq w \leq w_2$, is used to form the weighted combination of our two objectives). The size of the range $(w_2 - w_1)$ and its limiting elements (w_1, w_2) cannot

be known beforehand. The application of the weighting method must then be accomplished through a systematic, but arbitrary, scheme of varying the weighting factor w in an attempt to determine noninferior solutions. However, several runs of any algorithm using the weighted combination of objectives may produce the same solution. Further, should two successive runs with two different values of w, e.g., w_a & w_b, produce two different solutions, there exists no assurance that no other noninferior solutions exist which could be found if a weighting factor somewhere in between the two weights w_a and w_b were used. For example, at w_a = .1 Solution A is found, at w_b = .2 Solution B is found. There exists the possibility that, should another run using w = .15 or any other weight within the continuous interval (.1, .2) be accomplished, that some solution, Solution C, e.g., would be found. This is illustrated in Figure II-1. The analyst is left with the problem of establishing a weight-varying scheme to derive solutions and then, based on run results and his level of experience, adjust the scheme to search for other solutions. The final product is an approximate tradeoff curve with an unknown degree of accuracy.

The second difficulty lies with the inability of the weighting method to identify gap solutions. There does exist a multiobjective technique, the constraint method (see Cohon [97]), which addresses the problem of gap solutions. However, the constraint method is

FIGURE II-1

FINDING NONINFERIOR SOLUTIONS



Solution A found when $Z^w = 0.1\ Z_1 + (1 - 0.1)\ Z_2$.

Solution B found when $Z^w = 0.2\ Z_1 + (1 - 0.2)\ Z_2$.

Solution C found when $Z^w = 0.15\ Z_1 + (1 - 0.15)\ Z_2$.

Solutions D and E remain to be found.

directed toward linear programming problems and proceeds by altering the constraint set of such a problem. It is clearly inapplicable to shortest path labeling algorithms. It may, of course, be used with network flow algorithms adapted to determine shortest paths but the cost in time and effort may become extreme. Further, the selection of constraints is an ad-hoc process, much as the selection of weights in the weighting method.

Still another method exists, however. The next chapter offers a labeling algorithm which, quickly and relatively inexpensively, can generate the entire set of noninferior paths. The algorithm is unique in the field of multiobjective analysis in that complete generation of the noninferior set, including gap solutions, can be guaranteed in one run of the algorithm, without waste attributable the generation of repeated solutions.

CHAPTER III

THE NONINFERIOR PATH LABELING ALGORITHM

A. INTRODUCTION

The Noninferior Path Labeling Algorithm (NPLA), as its title suggests, is related in many ways to shortest path labeling algorithms. The particular class of shortest path labeling algorithms which bears on the discussion of NPLA is the set of label-correcting algorithms. In the discussion below, it will be shown that NPLA uses the structure and procedure of label-correcting algorithms while modifying the logic by which paths are constructed to enable determination of noninferior paths.

It was shown in Chapter II that the basic label-correcting algorithm proceeds through at most (N-1) iterations, where N is the number of nodes in the network. This is true since, at any iteration k, nodes are reached by paths consisting of k or less arcs. Because the longest simple path in an N-node network will contain (N-1) arcs, the algorithm need proceed no further. Another characteristic of the label-correcting algorithm is that all nodes, except the origin, will be reached and their node labels examined and altered as appropriate many times during the course of the algorithm's run.

These two factors are essential to the operation and convergence

of NPLA when based on label-correcting principles. NPLA modifies the label-correcting algorithm in three principle ways. First, arc weights consist not of a single number but of a P-tuple containing a separate number expressing the arc's contribution to each of P attributes of concern. Second, the node label is changed to be a label vector in which several labels, and predecessor indicators if desired, may be accepted. The third and most striking modification is the change from the minimization logic of the label-correcting algorithm to the logic of noninferiority. Throughout the operation of NPLA, attribute levels of the arcs, summations, and labels are maintained as distinct numbers within a P-tuple where P is the number of attributes.

A two-objective problem, with attributes A and B, can illustrate NPLA's operations. Each time a node is reached by a path during a run of NPLA, the attribute levels of the arc's 2 -tuples must be correspondingly added (i.e., using vector addition) to the one or more label 2-tuples of the connecting node. These summed 2-tuples, each representing a path, must be compared to the one or more label 2-triples of the reached node. The results of the comparison between any path 2-tuple and the label vector may take any of the following forms:

1) Both levels of attributes A and B in a path 2-tuple are better than both of the attribute levels of the labels in

the node vector. Here, dominance is exhibited and any dominated node label 2-tuples are eliminated from the node vector. Then the path 2-tuple is placed in the node vector. This is always the case the first time a node is reached by a path from the origin.

2) Both path attribute A and B are worse than the corresponding attribute levels of any label 2-tuple in the node vector. Here inferiority is exhibited. No change is made to the node vector and the path is dropped from further consideration.

3) Path attribute A (or B) is better than the corresponding attribute level of at least one of the label 2-tuples while path attribute B (or A) is worse than the corresponding attribute level of the label 2-tuple. Here, noninferiority is exhibited; the path cannot be considered either better or worse than at least one of the label 2-tuples. It should be noted that while a path may be noninferior to some of the 2-tuples, in the node vector, it may also dominate one or more of the remaining 2-tuples in which case actions outlined in paragraph 1), above are taken.

To illustrate the possible type of transaction in paragraph 3), an example is presented in Table III-1. Here, 2-tuples (8,4) and (6,6) in the node vector are dominated (we are minimizing both

TABLE III-1

SAMPLE NODE VECTOR VALUES

Path 2-tuple	Node Vector
	(4,10)
(5,2)	(6,6)
	(8,4)

objectives) by the path 2-tuple and must be removed. Path attribute A(5) is worse than the corresponding attribute level in the remaining node vector label 2-tuples (4,10). But, path attribute B(2) is better than the corresponding label attribute level. Noninferiority is exhibited and the path 2-tuple is included in the node vector. At the conclusion of the comparison process the node label vector would read as follows: Label vector [(4,10), (5,2)].

B. ALGORITHM STATEMENT

The algorithm may now be stated, with appropriate changes in notation and assuming a two-objective problem as an introduction.

Let, $d(i)$ = the node label vector associated with node i, the vector consisting of an initially undetermined number of label 2-tuples such that $d(i) = ((a_i^1, b_i^1), (a_i^2, b_i^2), \ldots, (a_i^r, b_i^r))$

$\ell(e_{ij})$ = (a_{ij}, b_{ij}), a 2-tuple expressing the arc's "length" in terms of the two attributes

1) Initially, $d(s) = (0,0)$

$d(i) = (\infty,\infty)$ $\quad i \neq s$

2) Search through the list of arcs for an arc e_{ij} such that: at least one of the sums of arc length and the current node labels at node i is dominant or noninferior with respect to the labels in the vector at node j. Label

vectors and arc lengths are summed as follows:

$$-((a_i^1, b_i^1), (a_i^2, b_i^2), \ldots, (a_i^r, b_i^r)) + (a_{ij}, b_{ij}) =$$

$$((a_i^1 + a_{ij}), (b_i^1 + b_{ij})), ((a_i^2 + a_{ij}), (b_i^2 + b_{ij})), \ldots,$$

$$((a_i^r, a_{ij}), (b_i^r + b_{ij}))$$

3) Continue the search procedure until no arcs can be found which will alter any node label or after (N-1) iterations. For each node j, d(j) contains the labels representing the two-dimensional lengths of all noninferior paths from s to j.

Several comments about the algorithm statement above are in order. There are several modifications to label-correcting shortest path algorithms, designed to increase efficiency, which can also be applied to NPLA. Discussion of some of these modifications is in Gilsinn and Witzgall [56] and Hulme and Wisniewski [72]. One of these modifications is the sequence list. Its basic concept is that, at any iteration, the number of calculations in step 2) can be reduced by the fact that once $\ell(e_{ij})$ has been used to check the node vector at j, e_{ij} need not be reexamined until a change occurs in the node vector at i. The sequence list and other modifications of this nature effect only the manner in which data is processed during the algorithm run. As such, modifications to shortest path algorithms can also be utilized with NPLA.

Another comment involves the concept of trees which was important in the discussion of shortest path labeling algorithms. NPLA does not construct a tree, except for the initial conditions and by accident in later iterations. Instead, what occurs is the simultaneous construction of a number of trees, of which one is a minimum tree. The use of the node vector preserves information about the network which would enable other trees to be constructed. The number of trees and how to construct them in a logical progression is believed to be more of a theoretical than practical question.

Bellman's principle of optimality no longer holds with NPLA. No longer are decisions at stages (iterations) being made according to the principle of optimality. However, the necessary substitution for the governing decision principle, necessary to obtain noninferior paths, can be stated in similar terms. It may be asserted that: a subpolicy of a noninferior policy must itself be noninferior. Thus, at no succeeding iteration can NPLA construct or build upon any but noninferior paths.

The final comment concerns NPLA's effectiveness. NPLA works by constructing, saving, and building upon noninferior paths during step 2) of the algorithm. Conversely, it will not construct, save, or build upon any inferior pathway. Further, the method of path construction does not consider the convexity or lack of convexity of the problem or its solution set, dealing only with node-arc

combinations. As a result, there exists no special obstacle to the discovery of gap solutions. Since they, too, can be noninferior, NPLA will find all noninferior solutions, including gap solutions as they exist.

There is no theoretical obstacle to using the NPLA on a problem of more than two objectives. However, there exist practical obstacles common to all techniques of multiobjective analysis. As the number of objectives increases, the number of comparison calculations in step 2) required to determine noninferiority increases. The number of noninferior paths discovered is likely to increase, requiring additional storage associated with the node vectors. The burden of display and subsequent use of results also increases with the number of objectives. In large problems or ones with a large number of objectives, the cost of implementing NPLA may be prohibitive. In such cases an estimating, e.g. weighting, approach may be more desirable.

While NPLA appears to be unique, there are elements of the algorithm for which the literature offers experience. The use of arc lengths of more than one attribute is well established. A comprehensive discussion of several problems employing this practice is in Christofides [62]. The use of what this paper calls node vectors, although derived independently, was later found to have been used in connection with D. Shier's approach to the K^{th} Shortest Path

Problem [79].

The alternatives to NPLA for a multiobjective approach to path-finding have been found to be rare in the literature. Christofides [62] simply mentions the use of weighted combinations of objectives. The K^{th} Shortest Path Problem appears to be the most touted, although faulted, approach among those associated with graph theory. Several examples of the use of multiobjective methodology in the analysis of network flow problems were found in the literature [93, 94, 98, 99, 100, 101]. The multiobjective techniques used in these examples are explained in the detailed examination of multiobjective methodology given in Cohon's 1978 book [97].

The result of literature search indicates that the Noninferior Path Labeling Algorithm is itself unique in its approach to the multiobjective path determination and represents a contribution to the field. Its practical effectiveness will be based on comparative efficiency with other multiobjective approaches. Preliminary investigations on sample networks have indicated that, with respect to sparse networks at least, the noninferior path algorithm is more efficient than the weighting method applied to shortest path labeling algorithms, even under best case assumptions applied to these latter methods. Firm conclusions await detailed research under a wide range of network conditions. But, sufficient confidence exists in its validity and general efficiency to have been used to develop the

path data for use in the DOE project.

C. NPLA SAMPLE PROBLEM

Figure III-1 gives a sample network which will be used to illustrate the NPLA procedure.

Problem: Find all noninferior paths from s to t

Initially, $d(s) = (0,0)$

$d(i) = (\infty,\infty)$ $\quad i \neq s$

At iteration 1), node vectors at 1 and 2 are the only ones which are subject to change.

$$d(s) + \ell(e_{s1}) = (0,0) + (2,4) = (2,4)$$

$$d(s) + \ell(e_{s2}) = (0,0) + (5,2) = (5,2)$$

$$d(i) + \ell(e_{i,j}) = (\infty,\infty) \quad \text{for all } i,j \neq s,1,2$$

Comparison of the above 2-tuples with the current node vectors at nodes 1 and 2 illustrates an obvious case of dominance and node vectors 1 and 2 are changed. At the conclusion of the first iteration, node vectors are as follows:

$d(s) = (0,0)$

$d(1) = (2,4)$

$d(2) = (5,2)$

$d(i) = (\infty,\infty)$ $\quad$ for $i \neq s,1,2$

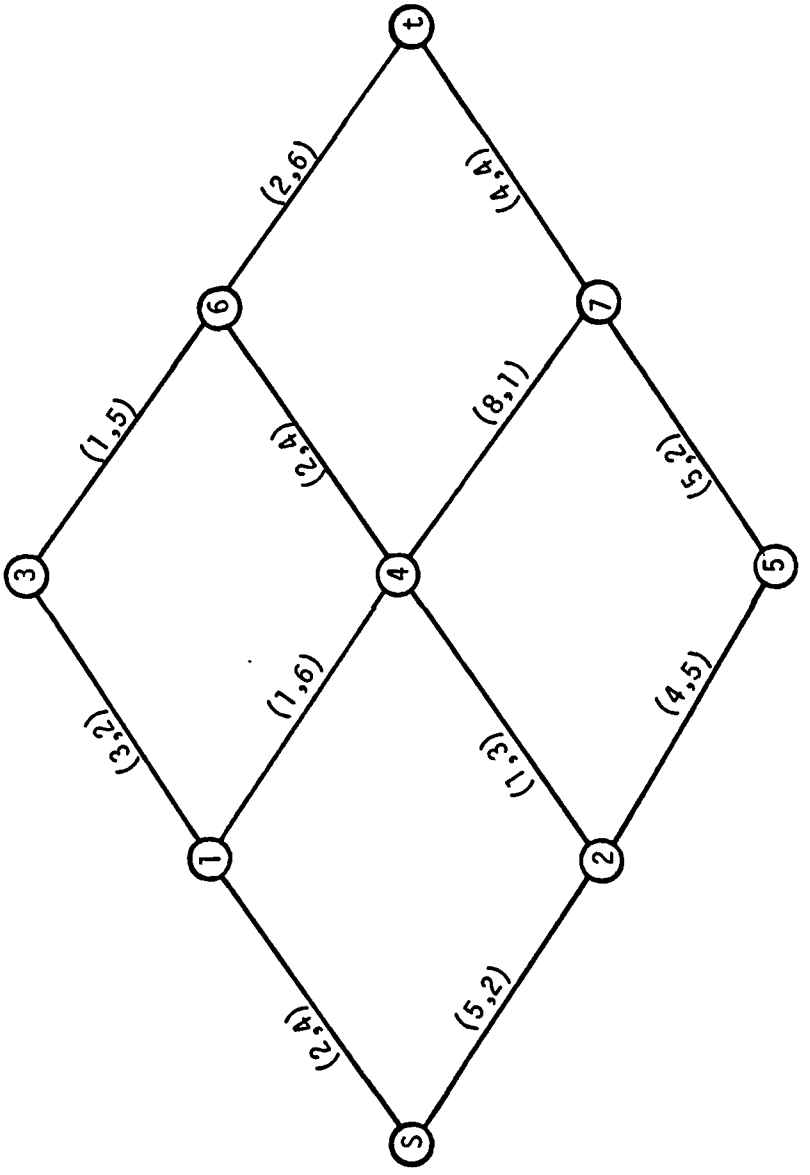


FIGURE III-1

SAMPLE NETWORK

Succeeding iterations proceed in a similar manner. Table III-2 illustrates the node vectors after each iteration of NPLA.

The algorithm terminates after the fifth iteration when no arc can be found which results in a change to any node vector. Four noninferior paths from s to t have been identified and their "lengths" recorded as labels in the node vector of node t. Note that, in a manner analogous to shortest path algorithms, all noninferior paths between s and all other nodes have also been identified. Paths may be retrieved, again as in shortest path algorithms, by maintaining node predecessor indicators with each label 2-tuple in the node vector. Such use is not required. The paths may be retrieved by finding the arc which, when its "length" is subtracted from a node label, results in the "value" of a node label at the predecessor node. For example, label (7,20) at node t is reduced by subtracting the length of arc e_{6t}, (2,6), to arrive at a value of (5,14). Since this corresponds to an actual label within the node vector of node 6, arc e_{6t} forms one link in a noninferior path from s to t. Similarly, one can trace the path back to the origin link by link from node 6. The choice of method when running NPLA on a computer is left to the programmer.

The noninferior paths, as represented by the label 2-tuples in the node vector of node t are:

TABLE III-2

NPLA RESULTS

Iteration	d(s)	d(1)	d(2)	d(3)	d(4)	d(5)	d(6)	d(7)	d(t)
1	(0,0)	(2,4)	(5,2)	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)	(∞,∞)
2	(0,0)	(2,4)	(5,2)	(5,6)	[(3,10), (6,5)]	(9,7)	(∞,∞)	(∞,∞)	(∞,∞)
3	(0,0)	(2,4)	[(4,13), (5,2)]	(5,6)	[(3,10, (6,5)]	[(8,18), (9,7)]	[(5,14), (6,11), (8,9)]	[(11,11), (14,6)]	(∞,∞)
4	(0,0)	(2,4)	[(4,13), (5,2)]	(5,6)	[(3,10), (6,5)]	[(8,18), (9,7)]	[(5,14), (6,11), (8,9)]	[(11,11), (14,6)]	[(7,20), (8,17), (10,15), (18,10)]
5	(0,0)	(2,4)	[(4,13), (5,2)]	(5,6)	[(3,10), (6,5)]	[(8,18), (9,7)]	[(5,14), (6,11), (8,9)]	[(11,11), (14,6)]	[(7,20), (8,17), (10,15), (18,10)]

$s \rightarrow 1 \rightarrow 4 \rightarrow 6 \rightarrow t$: (7,20)

$s \rightarrow 1 \rightarrow 3 \rightarrow 6 \rightarrow t$: (8,17)

$s \rightarrow 2 \rightarrow 4 \rightarrow 6 \rightarrow t$: (10,15)

$s \rightarrow 2 \rightarrow 4 \rightarrow 7 \rightarrow t$: (18,10)

These results may be checked by hand through construction of a decision tree. The complete enumeration of all paths from s to t in the sample network is made relatively easy given that only arc simple paths need be considered.

The tree for the example problem is given in Figure III-2. Twelve paths are identified by the decision tree. Ordering the path values by the first attribute level:

(7,20)

(8,17)

(10,15)

(13,24)

(15,15) → dominated by (10,15)

(17,24)

(18,10)

(18,13)

(20,20)

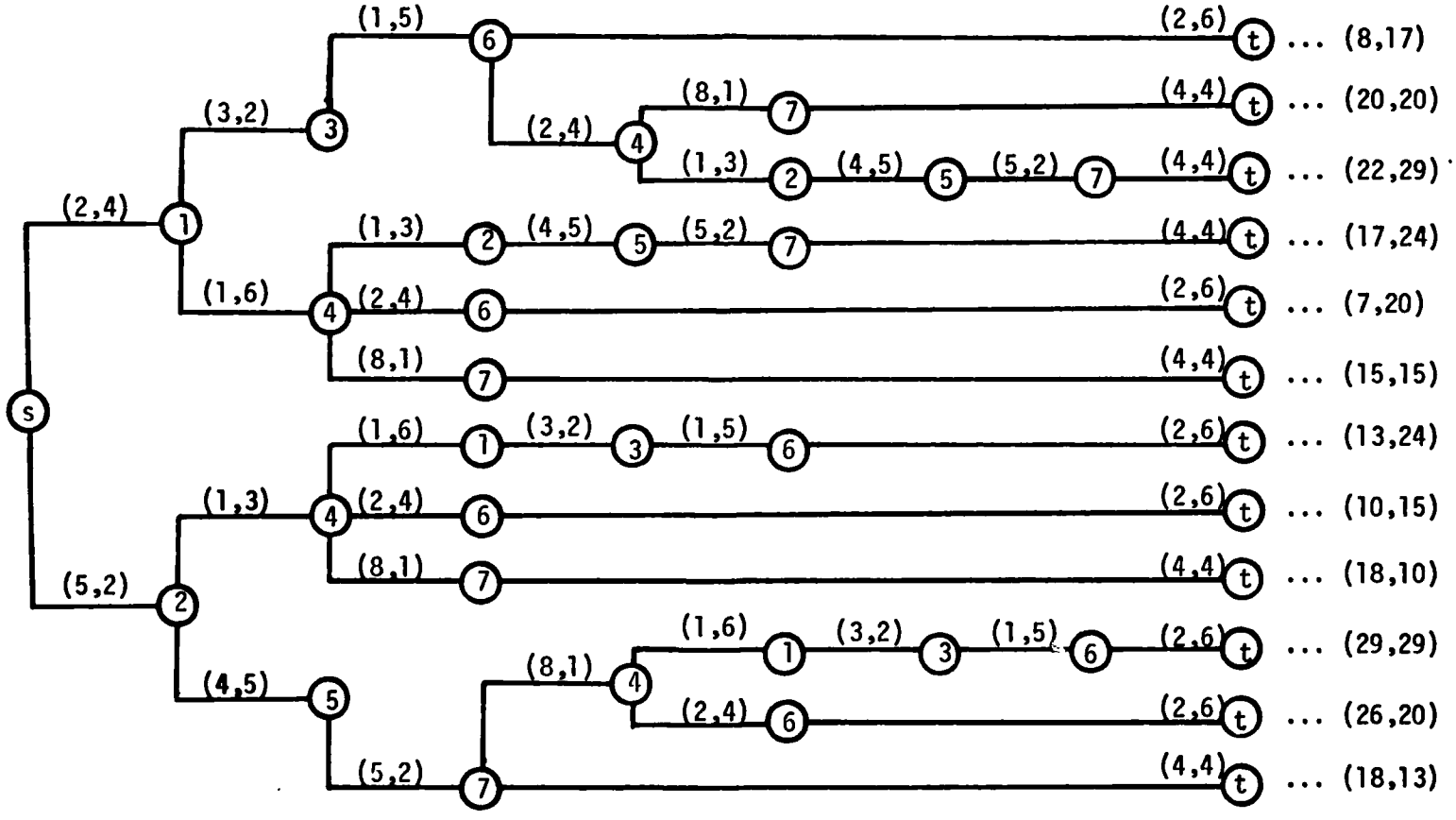
(22,29) → dominated by (18,10)

(26,20)

(29,29)

FIGURE III-2

SAMPLE PROBLEM DECISION TREE



The check for noninferiority is now a simple process of checking the values of the second attribute level. The list then reduces to [(7,20), (8,17), (10,15), (18,10)] which is identical, happily, to the node label vector of node t at the conclusion of NPLA. Note that only four of the twelve feasible paths are noninferior. Were a K Shortest Path algorithm to have been used to attempt to find these paths, it would have been required to set $k \geq 7$. Also, one would not be certain that all paths had been identified without the NPLA results or this exhaustive check.

The following chapter discusses the efficiency and effectiveness of NPLA. The results of NPLA's testing from the two-objective problem in the DOE project are used to provide concrete measures of these areas.

CHAPTER IV

NPLA APPLICATION RESULTS

A. COMPARATIVE PERFORMANCE

NPLA was used in the DOE project to generate sets of noninferior paths from potential AFR sites to commercial nuclear power plants in the eastern half of the continental United States. The network which formed the basis for the NPLA problem was a representation of the U.S. Interstate Highway System in the eastern half of the country. Some non-interstate routes were used, principally as connecting links between AFR or reactor sites and an interstate route. The set of nodes consisted of 75 reactor sites, 24 AFR sites, and 348 intersections on the highway system. Two attributes were used: measured road distance and population within a zone bordering along each side of the road. The objectives expressing these attributes were to be minimized.

The goal of the research team was to develop solutions to the AFR siting problem under varying assumptions about which reactors would require storage space for spent fuel and which AFR sites would be eligible to receive that fuel for storage. In all cases, the noninferior set was to be estimated. In some of the problems, it was desired to obtain the complete noninferior solution set. This latter desire demanded that the path determination problem, being embedded within the siting problem, also be solved to obtain the complete noninferior set of paths. Given that the noninferior set

of paths remains unaffected by the scope of the siting problem and the ease with which NPLA produced the noninferior path set, it was decided that NPLA be used to generate the complete noninferior set of paths from each of the 24 potential AFR sites to all reactors. This decision had the added benefit of producing consistency in the path sets and providing flexibility in the process of estimating the noninferior solution set of the siting problem.

NPLA was developed after a lengthy search for an appropriate path finding methodology. Prior to its development, three single objective algorithms were chosen for examination as bases for use with the weighting method of multiobjective analysis. One was the Out-of-Kilter Algorithm [68, 81] for which the network was formulated as a Minimum Cost Network Flow problem.

An excellent discussion of the Out-of-Kilter Algorithm is presented in Bazaara and Jarvis [68] and computational experience and comparison with other Minimum Cost Network Flow approaches is presented in Glover, et al., [87], among others [85, 91, 92]. As previously stated, labeling algorithms are far more efficient for finding shortest paths in uncapacitated networks. However, the Out-of-Kilter Algorithm was chosen as representative of the faster network flow algorithms.

The next two algorithms were shortest path labeling algorithms. The first was a modified label-correcting algorithm developed by project analyst Roger Cox. The last algorithm examined was

Dijkstra's label-setting shortest path algorithm [12] using a "heap sort" data list processing modification [56,72].

The resulting mix of algorithms presented a representative sample of the spectrum of available algorithms. Together they provided a check on consistency of results and added evidence of NPLA validity. They also provided a basis for the initial investigation into NPLA efficiency compared with alternate methods.

Table IV-1 presents the computational requirements for each of the four algorithms. Except for NPLA, the algorithms were run as single objective minimization problems using a weighted combination of the population and distance objectives. Each was run 9 times, 3 times from each of 3 AFR nodes as the origin. Times are averages for the 9 runs.

The results of the Out-of-Kilter Algorithm (OKA) show the time to achieve the solution to the basic problem. The results for the remaining algorithms include time to reconstruct and organize path data. Although this is contrary to the manner in which comparative analysis has been conducted in the literature, it offers an illustration of the computational times which an analyst ultimately faces in the practical use of a path-finding algorithm. Further, NPLA results include the time necessary to reconstruct those paths which represent solutions that lie on the convex hull of the solution set, but do not include time to reconstruct paths corresponding to gap solutions. While NPLA obtained these gap paths as part of its basic

TABLE IV-1

PATH GENERATION TIMES (SEC.)

Out-of-Kilter Algorithm (OKA)	Dijkstra Label-Setting Algorithm (DIJK)	Cox Label-Correcting Algorithm (LC)	Noninferior Path Labeling Algorithm (NPLA)
6.25	2.98	3.86	12.83

- All algorithms applied to 447 node network with 3 origins and 15 destinations.
- All runs accomplished during same time period on The Johns Hopkins University DEC 10 computer.

solution, omitting them from the reconstruction process served to place the results on a more even footing with those of the other algorithms which, as detailed earlier, are unable to find gap solutions.

The timing results illustrate that NPLA is on the order of 3.5 to 4.5 times slower than either of the single objective labeling algorithms. NPLA is seen to be on the order of 2 times slower than OKA. However, this difference effectively disappears given the subsequent effort needed to extract path data from OKA results.

Since NPLA produces the complete noninferior set in one run, the other algorithms must be further analyzed to provide a comparison based on equal production results. As a start, the results of NPLA can be examined. Table IV-2 illustrates the number of paths identified in 3 runs of NPLA using the 3 different AFR sites as the origin. Paths for each run were extracted from the origin to only 15 out of the 75, reactor sites. Again, only paths corresponding to solutions lying on the convex hull of the solution set were extracted. As a result, *a posteriori* assignment of weight intervals was possible. Each such weight interval corresponded to a path set, consisting of a path from the origin to each of the 15 destinations. Each such path set differed by at least one path from the path set associated with an adjacent weight interval. Each path set could be found by all of the other algorithms providing that a weight somewhere in each of the corresponding weight intervals were chosen to

TABLE IV-2

NPLA PATH GENERATION

Origin	No. of Destinations	No. of Weight Intervals (Distinct Path Sets)
A	15	22
B	15	29
C	15	24

TABLE IV-3

ESTIMATED EQUIVALENT PERFORMANCE TIMES (Sec.)

Origin	NPLA	DIJK	LC
A	8.94	70.11	85.65
B	13.49	84.97	119.38
C	16.07	67.17	85.52
Average	12.83	74.17	96.85

form the weighted combinations of objectives. Given n weight intervals identified by NPLA, each of the other algorithms could produce the same results if run n times with the proper weights chosen. Table IV-3 compares NPLA run times with equivalent performance run times for each of the other two labeling algorithms. OKA was eliminated from consideration based on previous results.

Under these "best case" calculations, NPLA is seen to be on the order of 4.2 to 9.6 times faster than the two labeling algorithms. However, this must be considered a lower bound on NPLA efficiency. Given the lack of prior knowledge of the number of path sets and their corresponding weight intervals, it would be extremely unusual for an analyst applying the weighting method to the single objective algorithms to arrive at a complete, exact weighting scheme to find the total solution set. It is most likely, as discussed earlier, that an initial weighting scheme followed by supplemental adjusted weighting schemes would be required, leading to a larger number of runs and a greater accumulated run time. Even then, the analyst would not be sure that all path sets had been identified without NPLA results as a check. Even then gap solutions still would remain undiscovered.

B. NPLA AND GAP SOLUTIONS

The issue of gap solutions is currently a subject of much interest in the literature. There have been several approaches

directed toward finding gap solutions [102, 103, 104] including the Constraint Method [See Cohon, 97]. Here, NPLA offers to increase the understanding of the nature and frequency of occurrence of gap solutions. Table IV-4 supplements the results from Table IV-2 with information about the number of distinct paths found which correspond to gap solutions and the solutions lying on the convex hull.

The number of gap solutions found was not trivial, a result which was surprising to the project team. It indicates that the solution sets of multiobjective discrete problems may be more complex than generally realized.

The results of this analysis conducted during the DOE project were deemed sufficient to justify use of NPLA for the solution of the path determination problem. Its use in similar problems appears also to be justified. The need and opportunity for further research on NPLA effectiveness and efficiency has been identified. The acceptance of NPLA methodology, while it appears certain, awaits the fulfillment of such research.

TABLE IV-4

ALL NONINFERIOR PATH GENERATION DATA

Origin	Path Sets	Convex Paths	Gap Paths	Total Paths
A	22	39	27	66
B	29	79	166	245
C	24	54	39	93

V. CONCLUSIONS

The Noninferior Path Labeling Algorithm has been presented as a new and effective approach to the multiobjective path-determination problem. In Chapter I, it was established that the problem which NPLA addresses is of growing importance to public and private planners. In Chapter II, NPLA was placed in historical context. Its shortest path labeling algorithm predecessors were examined and explained. Chapter II also examined the previous and alternative means of approaching the multiobjective path-determination problem and established NPLA as unique in its approach.

Chapter III formally presented NPLA and offered proof by analogy with its underlying approach — the shortest path label-correcting algorithm. An illustration of its functioning with a two-objective example was given. Finally, Chapter IV presented the initial evidence of the superiority of NPLA over alternative multiobjective techniques.

It is clear that NPLA promises to be a major contribution to the field of path-determination. It is also clear that further research on NPLA is both warranted and necessary. The following areas outline the general areas where future research should be rewarding.

A formal proof of NPLA efficacy is lacking. Its functional similarities to the shortest path label-correcting algorithm appear to promise that its proof would be equally similar, but this requires

verification. Along with a formal proof, theoretical bounds on computational efficiency should be investigated.

NPLA has been used only in conjunction with a single, sparse, positively weighted network. Its practical efficiency needs to be investigated with networks of varying size and arc density. Its use on networks with positive and negative arcs needs to be tested. Here again, its functional similarity to label-correcting algorithms promises that NPLA should work with such networks but this awaits verification. Comparative efficiency investigation needs to be more thoroughly and rigorously applied.

The practical efficiency of NPLA as used in the DOE project and reported in Chapter IV can most likely be improved. Improvements due to coding and the application of various list-processing modifications may lead to more efficient versions of NPLA, as they have done for shortest path algorithms.

NPLA, by the ease with which it identifies gap solutions, promises to add to the knowledge of the "duality gaps" exhibited in multiobjective path-determination problems and in many other discrete multiobjective problems. The number of gap solutions found by NPLA, as illustrated in Chapter IV, suggests to the author that a reevaluation of their significance is warranted.

Of final mention, all areas discussed above need to be evaluated for application of NPLA to problems of three or more objectives. Although the subject of limits to the use of NPLA in terms of computer

capacity may arise in the two objective case, it is under the condition of several objectives that NPLA is most likely to reach a practical limit on its effectiveness. What those limits are and where they arise need to be investigated.

Research on NPLA has just begun. The opportunity for further research is bountiful. While such research must follow, it can be said that NPLA has already been shown to be an effective and efficient approach to practical problems of multiobjective path-determination.

BIBLIOGRAPHY

A. SHORTEST PATH, DEVELOPMENT AND ALGORITHMS

1954

(1) Shimbel, A., "Structure in Communication Nets", *Proceedings of the Symposium on Information Networks*, N.Y., Brooklyn Polytechnic Institute (1954).

1956

(2) Ford, L.R., *Network Flow Theory*, Report P-923, The Rand Corporation, Santa Monica, Ca., August 1956. See also: 1958, Berge [(10), below] pp. 68-69.

(3) Kruskal, J.R., "On the Shortest Spanning Subtree of a Graph and the Travelling Salesman Problem", *Proceedings of the American Math. Soc'y.*, 1, 1956, pp. 48-50.

(4) Orden, A., "The Transshipment Problem", *Management Sci.*, 2(3), 1956, pp. 276-285.

1957

(5) Dantzig, G.B., "Discrete Variable Extremum Problems", *OPS Res.*, 5, Apr. 1957, pp. 266-277.

(6) Minty, G., "A Comment on the Shortest-Route Problem", *OPS Res.*, 5(5), 1957, p. 724.

(7) Moore, E.F., "The Shortest Path Through a Maze", *Proceedings of an International Symposium on the Theory of Switching*,

Part II, Apr. 1957. *In: Annals of the Computation Laboratory of Harvard University*, 30, Harvard University Press, Cambridge, Mass., pp. 285-292, 1959.

(8) Prim, R.C., Shortest Connection Networks and Some Generalizations, Bell System Technology Journal, pp. 1389-1401, Nov. 1957.

1958

(9) Bellman, R., "On a Routing Problem", *Quarterly of Applied Math*, 16, 1, Apr. 1958, pp. 87-90.

(10) Berge, C., *Theorie des Graphes et Ses Applications*, Dunod, Paris, 1958. In English, *Theory of Graphs*, Wiley, N.Y., 1962.

(11) Minty, G.J., "A Variant on the Shortest Route Problem", *OPS Res.*, 6, 1958, p. 882.

1959

(12) Dijkstra, E.W., "A Note on Two Problems in Connection with Graphs", *Numerische Mathematik*, 1, 1959, pp. 269-271.

(13) Hoffman, W. and R. Pavley, "A Method for Solution of the N^{th} Best Path Problem", *Journal of the Association of Computing Machinery, (A.C.M.)*, 6, 1959, pp. 506-514.

1960

(14) Bellman, R. and R. Kalaba, "On K^{th} Best Policies", *Journal of S.I.A.M.*, 8, 1960, pp. 582-588.

(15) Dantzig, G., "On the Shortest Path Through a Network",

Management Sci., 6, 1960, pp. 187-190.

(16) Peart, R.M., P.H. Randolph, and T.E. Bartlett, "The Shortest Route Problem", *OPS Res.*, 8(6), 1960, pp. 866-868.

(17) Pollack, M., and W. Wiebenson, "Solutions of the Shortest Route Problem - A Review", *OPS Res.*, 8(2), 1960, pp. 224-230.

(18) Whiting, P.D. and J.A. Hillier, "A Method For Finding the Shortest Route Through a Road Network", *Operational Res. Quar.*, 11(1-2), 1960, pp. 37-40.

1961

(19) Pollack, M., "The K^{th} Best Route Through a Network", *OPS Res.*, 9, 1961, pp. 578-579.

(20) Pollack, M., "Solutions of the K^{th} Best Route Through a Network - A Review", *Journal of Mathematical Analysis and Application*, 3, 1961, pp. 547-559.

(21) Pollack, M. and W. Wiebenson, "Comments on 'The Shortest Route Problem' by Peart, Randolph, and Bartlett", *OPS. Res.*, 9, 1961, pp. 411-412.

1962

(22) Floyd, R.W., "Algorithm 97, Shortest Path", *Communication of A.C.M.*, 5(6), 1962, p. 345.

(23) Ford, L.R. and D. R. Fulkerson, *Flows in Networks*, Princeton University Press, Princeton, N.J., 1962.

(24) Mintz, G.J., "On An Algorithm for Solving Some Network

Programming Problems", *OPS Res.*, 10(3), May-June 1962, pp. 403-405.

1963

(25) Clarke, S., R. Krikorian, and J. Rausen, "Computing the N Best Loopless Paths in a Network", *S.I.A.M.*, 11, 1963, pp. 1096-1102.

(26) Kolker, R., D. Liss, and S. Okada, *Algebraic Techniques of Path Finding and Minimum Path Finding in Graphs*, TM 3421, Mitre Corp., Bedford, Mass., 1963.

1964

(27) Klee, V.L., "A String Algorithm for Shortest Paths in a Directed Network", *OPS Res.*, 12(3), May-June 1964, pp. 428-432.

1965

(28) Berge, C. and A. Ghouila-Houri, *Programming Games and Transportation Networks*, John Wiley and Sons, N.Y., 1965.

(29) Busacker, R.G. and T.L. Saaty, *Finite Graphs and Networks: An Introduction with Applications*, McGraw-Hill, N.Y., 1965.

(30) Farbey, B.A., A.H. Land, and J.D. Murchland, *The Cascade Algorithm For Finding All Shortest Distances in a Directed Graph*, Report LSE-TNT-19, Transportation Network Theory Unit, London School of Economics (Later of the London School of Business Studies), 28 Northumberland Avenue,

London, WC2, 1965. Published in *Management Sci.*, 14(1), Sept. 1967, pp. 19-28.

(31) Johnson, E.L., *Programming in Networks and Graphs*, Res. Rept. ORC 65-1, Univ. of Ca., Berkeley, Ca., 1965.

(32) Murchland, J.D., *A New Method for Finding All Elementary Paths in a Complete Directed Graph*, Rept. LSE-TNT-22, Transport Network Theory Unit, London School of Economics, London, Oct. 1965.

1966

(33) Dantzig, G.B., *All Shortest Paths in a Graph*, Tech. Rept. 66-3, Operations Research House, Stanford Univ., Stanford, Ca., Nov. 1966. Also: pp. 92-92, *Theory of Graphs: International Symposium, Rome, July 1966*, P. Rosenstiehl, Ed., Gordon and Breach, N.Y., 1967.

(34) Dantzig, G.B., W.D. Blattner, and M.R. Rao, *All Shortest Routes Routes from a Fixed Origin in a Graph*, Tech. Report 66-2, Operations Research House, Stanford University, Stanford, Ca., Nov. 1966. Also: pp. 85-90, *Theory of Graphs: International Symposium, Rome, July 1966*, P. Rosenstiehl, Ed., Gordon and Breach, N.Y., 1967.

(35) Fulkerson, D.R., "Flow Networks and Combinatorial Operations Research", *The Amer. Mathematical Monthly*, 73(2), Feb. 1966, pp. 115-138.

(36) Johnson, E.L., "Networks and Basic Solutions", *OPS Res.*, 14, 1966, pp. 619-623.

(37) Hu, T.C., *A Decomposition Algorithm for Shortest Paths in a Network*, Res. Paper RC-1562, IBM Watson Research Center, Yorktown Hts, N.Y., 1966. See later version: *OPS Res.*, 16(1), Jan-Feb. 1968, pp. 91-102.

(38) Mills, G., "A Decomposition Algorithm for the Shortest Route Problem", *OPS Res.*, 14(2), Mar-Apr. 1966, pp. 279-291.

(39) Nicholson, T.A.J., "Finding the Shortest Route Between Two Points in a Network", *The Computer Journal*, 9(3), Nov. 1966, pp. 275-280.

(40) Sakarovitch, M., *The K Shortest Routes and K Shortest Chains in a Graph*, Rept. ORC 66-32, Operations Research Center, Univ. of California, Berkeley, Ca., Oct. 1966. Also: *Trans. Res.*, 2(1), Mar. 1968, pp. 1-11.

(41) Saksena, J. and S. Kumar, "The Routing Problem with 'K' Specified Nodes", *OPS Res.*, 14, 1966, pp. 909-913.

1967

(42) Land, A.H. and S.W. Stairs, "An Extension of the Cascade Algorithm to Large Graphs", *Management Sci.*, 14(1), Sept., 1967, pp. 29-33.

1968

(43) Hitchner, L.E., *A Comparative Investigation of the Computational*

Efficiency of Shortest Path Algorithms, Rept. ORC-68-25, Operations Research Center, Univ. of California, Berkeley, Ca., 1968.

1969

(44) Dreyfus, S.E., "An Appraisal of Some Shortest Path Algorithms", *OPS Res.*, 17(3), 1969, pp. 395-412.

(45) Hu, T.C., *Integer Programming and Network Flows*, Addison-Wesley, Reading, Mass., 1969.

(46) Murchland, J.D., *A Bibliography of the Shortest Route Problem*, Rept. LBS-TNT-6.2, Transport Network Theory Unit, London Graduate School of Business Studies, 28 Northumberland Ave., London WC-2, 1969.

1970

(47) Yen, J., "An Algorithm for Finding Shortest Routes from all Source Nodes to a Given Destination in General Networks", *Quart. of App. Math.*, 27, 1970, pp. 526-530.

1971

(48) Braess, D., "The Computation of the Shortest Path in Graphs and Appropriate Data Structures", (in German) *Computing*, 8, 1971, pp. 171-181.

(49) Frank, H. and I. Frisch, *Communication, Transmission, and Transportation Networks*, Addison-Wesley, Reading, Mass., 1971.

(50) Marshall, C.W., *Applied Graph Theory*, Wiley-Interscience, N.Y. 1971.

(51) Price, W.L., *Graphs and Networks*, Auerbach Publishers, Inc., Princeton, N.J., 1971.

1972

(52) Johnson, E., "On Shortest Paths and Sorting", *Proceedings of 1972 ACM Conference*, Boston, Aug. 1972, pp. 510-517.

(53) Nemhauser, G., "A Generalized Permanent Label Setting Algorithm for the Shortest Path Between Specified Nodes", *Jour. of Math. Anal. and Appl.*, 38, 1972, pp. 328-334.

(54) Potts, R.B. and R.M. Oliver, *Flows in Transportation Networks*, Academic Press, N.Y., 1972.

(55) Yen, J.Y., "Finding the Length of all Shortest Paths in N-Node Nonnegative Distance Complete Networks Using 1/2 N^3 Additions and N^3 Comparisons", *Jour. ACM*, 19(3), 1972, pp. 423-424.

1973

(56) Gilsinn, J. and C. Witzgall, *A Performance Comparison of Labeling Algorithms for Calculating Shortest Path Trees*, Tech. Note 772, National Bureau of Standards, GPO, Washington, D.C., May 1973.

(57) Spira, P.M., "New Algorithm for Finding All Shortest Paths in a Graph of Positive Arcs in Average Time in the Order of $(N^2) \cdot (\text{LOG } N)^2$", *S.I.A.M. Jour. of Computing*, 2(1), Mar. 1973, pp. 28-32.

(58) Tabourier, Y., "All Shortest Distances in a Graph", *Discrete Math*, 4, 1973, pp. 83-87.

1974

(59) Bazaraa, M. and R.W. Langley, "A Dual Shortest Path Algorithm", *S.I.A.M. APP Math*, 26(3), May 1974, pp. 496-501.

(60) Pape, U., "Implementation and Efficiency of Moore-Algorithms for the Shortest Route Problem", *Math. Prog.*, 7, 1974, pp. 212-222.

(61) Shier, D., "Computational Experience with an Algorithm for Finding the K Shortest Paths in a Network", *Jour. of Res. of NBS*, 78B, 1974, pp. 139-165.

1975

(62) Christofides, N., *Graph Theory: An Algorithmic Approach*, Academic Press, N.Y., 1975.

(63) Pierce, A.R., "Bibliography on Algorithms for Shortest Path, Shortest Spanning Tree, and Related Circuit Routing Problems (1956-74)", *Networks*, 5(2), April 1975, pp. 129-150.

1976

(64) Golden, B., "Shortest Path Algorithms: A Comparison", *OPS Res.*, 24(6), 1976, pp. 1164-1168.

(65) Shier, D., "Iterative Methods for Determining the K Shortest Paths in a Network", *Networks*, 6(3), 1976, pp. 205-230.

(66) Thomas, R., "On Optimum Path Problems", *Networks*, 6(4), 1976, pp. 287-306.

(67) Wagner, R., "A Shortest Path Algorithm for Edge-Sparse Graphs", *J.A.C.M.*, 23(1), 1976, pp. 50-57.

1977

(68) Bazaraa, M.S. and J.J. Jarvis, *Linear Programming and Network Flows*, John Wiley and Sons, N.Y., 1977.

(69) Dreyfus, S.E. and A.M. Law, *The Art and Theory of Dynamic Programming*, Academic Press, N.Y., 1977.

(70) Golden, B.L. and T.L. Magnanti, "Deterministic Network Optimization: A Bibliography", *Networks*, 7(2), 1977, pp. 149-184.

(71) Johnson, D.B., "Efficient Algorithm for Shortest Paths in Sparse Networks", *J.A.C.M.*, 24(1), 1977, pp. 1-13.

1978

(72) Hulme, B.L. and J.A. Wisniewski, *A Comparison of Shortest Path*

Algorithms Applied to Sparse Graphs, SAND 78-1411

(73) Kelton, W.D. AND A.M. Law, "A Mean-Time Comparison of Algorithms for the All-Pairs Shortest-Path Problem with Arbitrary Arc Lengths", *Networks*, 8(2), 1978, pp. 97-106.

(74) Minieka, E., *Optimization Algorithms for Networks and Graphs*, Marcel Dekker, Inc., N.Y., 1978.

(75) VanVliet, D., "Improved Shortest Path Algorithms for Transport Networks", *Trans. Res.*, 12, 1978, pp. 7-20.

1979

(76) Denardo, E.V. and B.L. Fox, "Shortest-Route Methods: 1. Reaching, Pruning, and Buckets", *OPS Res.*, 27(1), Jan-Feb. 1979, pp. 161-186.

(77) Denardo, E.V. and B.L. Fox, "Shortest-Route Methods: 2. Group Knapsacks, Expanded Networks, and Branch-and-Bound", *OPS Res.*, 27(3), May-June 1979, pp. 548-566.

(78) Dial, R., F. Glover, D. Karney, and D. Klingman, "A Computational Analysis of Alternative Algorithms and Labeling Techniques for Finding Shortest Path Trees", *Networks*, 9(3), 1979, pp. 215-248.

(79) Shier, D., "On Algorithms for Finding the K Shortest Paths in a Network", *Networks*, 9(3), 1979, pp. 195-214.

B. NETWORK FLOW ALGORITHMS

1957

(80) Ford, L.R. and D.R. Fulkerson, "A Primal-Dual Algorithm for the Capacitated Hitchcock Problem", *Nav. Res. Log Quart.*, 4, 1957, pp. 47-54.

1961

(81) Fulkerson, D.R., "An Out-of-Kilter Method for Solving Minimal Cost Flow Problems", *S.I.A.M. Jour. of App. Math*, 9(1), 1961, pp. 18-27.

1967

(82) Klein, M., "A Primal Method for Minimal Cost Flows", *Management Sci.*, 14(3), Nov. 1967, pp. 205-220.

1972

(83) Glover, F., D. Klingman, and A. Napier, "An Efficient Dual Approach to Network Problems", *OPS Res.*, 9(1), 1972, pp. 1-18.

1973

(84) Srinivasen, V. and G.L. Thompson, "Benefit-Cost Analysis of Coding Techniques for the Primal Transportation Algorithm", *Jour. of the Assoc. for Computing Machinery*, 20, 1973, pp. 194-213.

1974

(85) Barr, R.S., F. Glover, and D. Klingman, "An Improved Version of the Out-of-Kilter Method and a Comparative Study of Computer Codes", *Mathematical Programming*, 7(1), 1974, pp. 60-86.

(86) Glover, F., D. Karney, D. Klingman, and A. Napier, "A Computational Study on Start Procedures, Basis Change Criteria, and Solution Algorithms for Transportation Problems", *Management Sci.*, 20(5), 1974, pp. 793-813.

(87) Glover, F., D. Karney, and D. Klingman, "Implementation and Computational Comparisons of Primal, Dual, and Primal-Dual Computer Codes for Minimum Cost Network Flow Approaches", *Networks*, 4(3), 1974, pp. 191-212.

1975

(88) Charnes, A., F. Glover, D. Karney, D. Klingman, and J. Stutz, "Past, Present, and Future of Development, Computational Efficiency, and Practical Use of Large Scale Transportation and Transshipment Computer Codes", *Computers and OPS Res.*, 2, 1975, pp. 71-81.

1977

(89) Bradley, G.H., G.G. Brown, and G.W. Graves, "Design and Implementation of Large Scale Primal Transshipment Algorithms", *Management Sci.*, 24(1), 1977, pp. 1-34.

(90) Chen, S. and R. Saigal, "A Primal Algorithm for Solving A Capacitated Network Flow Problem with Additional Linear Constraints", *Networks*, 7(1), 1977, pp. 59-80.

(91) Shapiro, J.F., "A Note on the Primal-Dual and Out-of-Kilter Algorithms for Network Optimization Problems", *Networks*, 7(1), 1977, pp. 81-88.

1978

(92) Ali, A.I., R.V. Helgason, J.L. Kennington, and H.S. Lall, "Primal Simplex Network Codes: State-of-the-Art Implementation Technology", *Networks*, 8(4), 1978, pp. 315-339.

C. MULTIOBJECTIVE ANALYSIS AND RELATED TOPICS

1964

(93) Elmaghraby, S.E., "Sensitivity Analysis of Multi-Terminal Flow Networks", *OPS Res.*, 12(5), Sept.-Oct. 1964, pp. 680-688.

1971

(94) Ruffli, T., "Decentralized Transshipment Networks", *OPS Res.*, 19(7), 1971, pp. 1619-1631.

1977

(95) Adulbhan, P., and M.T. Tabucanon, "Bicriterion Linear Programming", *Computers and OPS Res.*, 4(2), 1977, pp. 147-153.

(96) Kornbluth, J.S.H., "The Fuzzy Dual: Information for the

Multiple Objective Decision-Maker", *Computing and OPS Res.*, 4(1), 1977, pp. 65-72.

1978

(97) Cohon, J., *Multiobjective Programming*, Academic Press, N.Y., 1978.

(98) Moore, L., B.W Taylor, III, and S.M. Lee, "Analysis of a Transshipment Problem with Multiple Conflicting Objectives", *Computing and OPS Res.*, 5(1), 1978, pp. 39-46.

1979

(99) Aneja, Y.P. and K.P.K. Nair, "Bicriteria Transportation Problem", *Management Sci.*, 25(1), Jan. 1979, pp. 73-79.

(100) Sharma, J.K. and K. Swarup, "Time-Cost Tradeoff in a Multidimensional Transportation Problem-I", *OPS Res.*, 16(1), 1979, pp. 23-33.

(101) Stewart, T.S. and H.W. Ittman, "Two-Stage Optimization in a Transportation Problem", *Jour. of the Operational Res. Soc.* (U.K.), 30(10), Oct. 1979, pp. 897-904.

D. DUALITY GAPS

1969

(102) Greenberg, H.J., "Lagrangian Duality Gaps: Their Source and Resolution", Tech. Rept. CP-69005, Southern Methodist University, Dallas, Texas, 1969.

1975

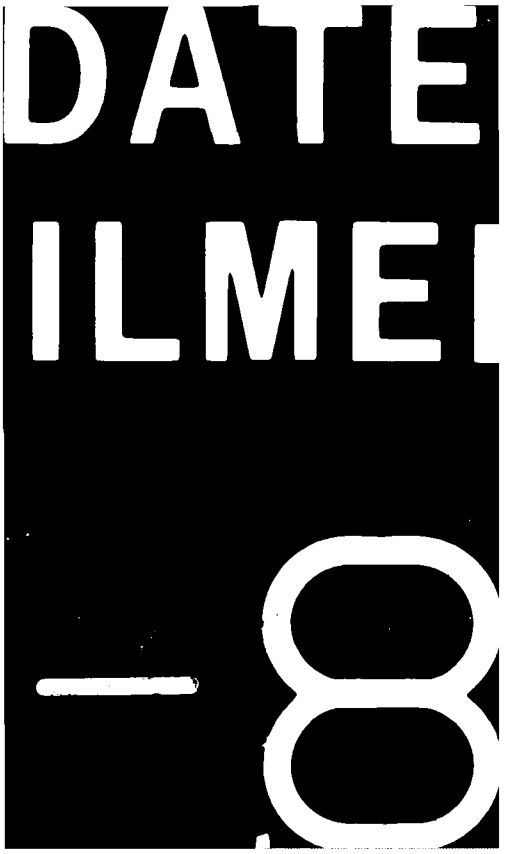
(103) Rossman, L.A., "Resolution of Duality Gaps with Dynamic Programming", Presented at ORSA/TIMS Joint National Meeting, Nov. 17-19, Las Vegas, Nevada, 1975.

(104) Stephanopoulos, G. and A.W. Westerborg, "The Use of Hestenes' Method of Multipliers to Resolve Dual Gaps in Engineering System Optimization", *J.O.T.A.*, 15, 1975, pp. 285-309.

VITA

Joseph E. Thomas, Jr. was born on July 10, 1949 in Philadelphia, Pennsylvania. His family moved to Erie, Pennsylvania in 1950 and to Parma, Ohio in 1958 where he attended Parma High School, graduating in 1967. Mr. Thomas spent four years at the United States Military Academy at West Point, New York and graduated in June 1971 with a Bachelor of Science degree. Following graduation, Mr. Thomas was commissioned as a second lieutenant in the United States Army. He served as a rifle platoon leader and intelligence officer with the 25th Infantry Division at Schofield Barracks, Hawaii from 1971 to 1974. In 1975, he was promoted to the rank of captain. From 1975 to 1977, he served as an intelligence officer with the United States Army Intelligence Agency at Fort George G. Meade, Maryland. From 1977 to 1978, Mr. Thomas attended the Defense Intelligence School in Washington, D.C. He entered the Department of Geography and Environmental Engineering at The Johns Hopkins University in September 1978 and completed the requirements for the Master of Science in Engineering degree in June 1980.

Mr. Thomas married his wife, Ann, in September 1973. The Thomas's daughter, Christine was born in October 1974. In addition to his military career, Mr. Thomas has interests in history, political science, and athletics.


DATE
ILME
-8